**Annex W. Queries to extract features. Approach to data set creation.**

/\* Query activities and events of the students of the group specified (course level) \*/

SELECT

l.userid,

l.eventname,

l.action,

l.target,

l.timecreated AS start\_time,

CASE

WHEN l.timecompleted IS NOT NULL THEN l.timecompleted

ELSE l.timecreated

END AS end\_time,

TIMESTAMPDIFF(

SECOND,

FROM\_UNIXTIME(l.timecreated),

FROM\_UNIXTIME(l.timecreated),

~~/\* FROM\_UNIXTIME(~~

~~CASE~~

~~WHEN l.timecompleted IS NOT NULL THEN l.timecompleted~~

~~ELSE l.timecreated~~

~~END~~

~~\*/~~

)

) AS duration\_seconds,

cm.instance AS activity\_id,

m.name AS activity\_name,

c.fullname AS course\_name

FROM

mdl\_logstore\_standard\_log l

JOIN

mdl\_course\_modules cm ON l.contextinstanceid = cm.id

JOIN

mdl\_modules m ON cm.module = m.id

JOIN

mdl\_course c ON cm.course = c.id

WHERE

l.userid IN (

SELECT u.id

FROM mdl\_user u

JOIN mdl\_user\_enrolments ue ON ue.userid = u.id

JOIN mdl\_enrol e ON e.id = ue.enrolid

JOIN mdl\_course c ON c.id = e.courseid

JOIN mdl\_role\_assignments ra ON ra.userid = u.id

JOIN mdl\_context ctx ON ctx.id = ra.contextid

JOIN mdl\_role r ON r.id = ra.roleid

WHERE u.department = 'ITfu-23'

AND r.shortname = 'student'

AND ctx.contextlevel = 50

)

AND l.eventname != ''

AND l.timecreated > 0

ORDER BY

l.timecreated ASC;

/\*The **duration between events**, where each event's duration is the **difference between its timestamp and the next event's timestamp** (for the same user). This is often called a "lead" time calculation:

* LEAD() is used to get the timestamp of the next event per user.
* PARTITION BY l.userid makes sure the duration is calculated per student.
* If LEAD() is not supported in your DB version, I can rewrite it using a correlated subquery (less efficient, but compatible).\*/

SELECT

l.userid,

l.eventname,

l.action,

l.target,

l.timecreated AS start\_time,

LEAD(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated) AS next\_event\_time,

TIMESTAMPDIFF(

SECOND,

FROM\_UNIXTIME(l.timecreated),

FROM\_UNIXTIME(

LEAD(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated)

)

) AS duration\_seconds,

cm.instance AS activity\_id,

m.name AS activity\_name,

c.fullname AS course\_name

FROM

mdl\_logstore\_standard\_log l

JOIN

mdl\_course\_modules cm ON l.contextinstanceid = cm.id

JOIN

mdl\_modules m ON cm.module = m.id

JOIN

mdl\_course c ON cm.course = c.id

WHERE

l.userid IN (

SELECT u.id

FROM mdl\_user u

JOIN mdl\_user\_enrolments ue ON ue.userid = u.id

JOIN mdl\_enrol e ON e.id = ue.enrolid

JOIN mdl\_course c ON c.id = e.courseid

JOIN mdl\_role\_assignments ra ON ra.userid = u.id

JOIN mdl\_context ctx ON ctx.id = ra.contextid

JOIN mdl\_role r ON r.id = ra.roleid

WHERE u.department = 'ITfu-23'

AND r.shortname = 'student'

AND ctx.contextlevel = 50

)

AND l.eventname != ''

AND l.timecreated > 0

ORDER BY

l.userid,

l.timecreated ASC;

/\* Group Moodle log events into **sessions based on idle gaps** (e.g., 60 minutes or 3600 seconds (group into sessions based on idle gaps)): the logic compares each event's timestamp to the previous event for the same user and increments a **session counter** when the time gap exceeds the threshold \*/

WITH ordered\_logs AS (

SELECT

l.userid,

l.eventname,

l.action,

l.target,

l.timecreated,

cm.instance AS activity\_id,

m.name AS activity\_name,

c.fullname AS course\_name,

LAG(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated) AS prev\_time

FROM

mdl\_logstore\_standard\_log l

JOIN

mdl\_course\_modules cm ON l.contextinstanceid = cm.id

JOIN

mdl\_modules m ON cm.module = m.id

JOIN

mdl\_course c ON cm.course = c.id

WHERE

l.userid IN (

SELECT u.id

FROM mdl\_user u

JOIN mdl\_user\_enrolments ue ON ue.userid = u.id

JOIN mdl\_enrol e ON e.id = ue.enrolid

JOIN mdl\_course c ON c.id = e.courseid

JOIN mdl\_groups\_members gm ON gm.userid = u.id

JOIN mdl\_groups g ON g.id = gm.groupid

JOIN mdl\_role\_assignments ra ON ra.userid = u.id

JOIN mdl\_context ctx ON ctx.id = ra.contextid

JOIN mdl\_role r ON r.id = ra.roleid

WHERE u.department = 'ITfu-23'

AND r.shortname = 'student'

AND ctx.contextlevel = 50

)

AND l.eventname != ''

AND l.timecreated > 0

),

session\_flagged AS (

SELECT \*,

CASE

WHEN prev\_time IS NULL THEN 1

WHEN timecreated - prev\_time > 1800 THEN 1

ELSE 0

END AS new\_session

FROM ordered\_logs

),

session\_numbered AS (

SELECT \*,

SUM(new\_session) OVER (PARTITION BY userid ORDER BY timecreated) AS session\_id

FROM session\_flagged

)

SELECT

userid,

session\_id,

eventname,

action,

target,

timecreated,

activity\_id,

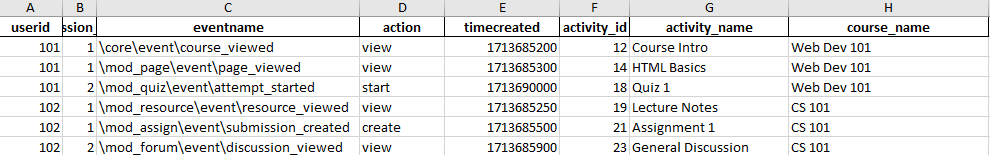
activity\_name,

course\_name

FROM session\_numbered

ORDER BY userid, session\_id, timecreated;

Example of data to be returned:



/\*with duration and grouping by session - the query will return results that include:

* **userid**: The ID of the user.
* **eventname**: The name of the event.
* **action**: The action associated with the event.
* **target**: The target associated with the event.
* **start\_time**: The timestamp when the event started.
* **next\_event\_time**: The timestamp of the next event for the same user.
* **duration\_seconds**: The duration of the event in seconds.
* **activity\_id**: The ID of the activity that the event is related to.
* **activity\_name**: The name of the activity.
* **course\_name**: The name of the course.
* **session\_id**: The session ID, which groups events based on idle gaps exceeding 3600 seconds.

\*/

WITH EventSessions AS (

SELECT

l.userid,

l.eventname,

l.action,

l.target,

l.timecreated AS start\_time,

LEAD(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated) AS next\_event\_time,

TIMESTAMPDIFF(

SECOND,

FROM\_UNIXTIME(l.timecreated),

FROM\_UNIXTIME(

LEAD(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated)

)

) AS duration\_seconds,

cm.instance AS activity\_id,

m.name AS activity\_name,

c.fullname AS course\_name,

-- Generate session ID based on idle gaps greater than 3600 seconds (60 minutes)

SUM(

CASE

WHEN TIMESTAMPDIFF(SECOND,

FROM\_UNIXTIME(LAG(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated)),

FROM\_UNIXTIME(l.timecreated)) > 3600 THEN 1

ELSE 0

END

) OVER (PARTITION BY l.userid ORDER BY l.timecreated) + 1 AS session\_id

FROM

mdl\_logstore\_standard\_log l

JOIN

mdl\_course\_modules cm ON l.contextinstanceid = cm.id

JOIN

mdl\_modules m ON cm.module = m.id

JOIN

mdl\_course c ON cm.course = c.id

WHERE

l.userid IN (

SELECT u.id

FROM mdl\_user u

JOIN mdl\_user\_enrolments ue ON ue.userid = u.id

JOIN mdl\_enrol e ON e.id = ue.enrolid

JOIN mdl\_course c ON c.id = e.courseid

JOIN mdl\_role\_assignments ra ON ra.userid = u.id

JOIN mdl\_context ctx ON ctx.id = ra.contextid

JOIN mdl\_role r ON r.id = ra.roleid

WHERE u.department = 'ITfu-23'

AND r.shortname = 'student'

AND ctx.contextlevel = 50

)

AND l.eventname != ''

AND l.timecreated > 0

)

SELECT

es.userid,

es.eventname,

es.action,

es.target,

es.start\_time,

es.next\_event\_time,

es.duration\_seconds,

es.activity\_id,

es.activity\_name,

es.course\_name,

es.session\_id

FROM EventSessions es

ORDER BY

es.userid,

es.session\_id,

es.start\_time ASC;

Example of data to be returned:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **userid** | **eventname** | **action** | **target** | **start\_time** | **next\_event\_time** | **duration\_seconds** | **activity\_id** | **activity\_name** | **course\_name** | **session\_id** | | 1 | course\_viewed | view | course | 2025-04-20 10:00 | 2025-04-20 10:10 | 600 | 101 | Quiz | Introduction to Programming | 1 | | 1 | quiz\_attempted | attempt | quiz | 2025-04-20 10:10 | 2025-04-20 10:15 | 300 | 102 | Quiz | Introduction to Programming | 1 | | 1 | course\_viewed | view | course | 2025-04-20 11:20 | 2025-04-20 11:25 | 300 | 103 | Assignment | Advanced Programming | 2 | | 1 | quiz\_attempted | attempt | quiz | 2025-04-20 11:25 | 2025-04-20 11:30 | 300 | 104 | Quiz | Advanced Programming | 2 | | 2 | course\_viewed | view | course | 2025-04-20 09:00 | 2025-04-20 09:30 | 1800 | 105 | Forum | Web Development | 1 | | 2 | forum\_posted | post | forum | 2025-04-20 09:30 | 2025-04-20 09:35 | 300 | 106 | Forum | Web Development | 1 | | 2 | course\_viewed | view | course | 2025-04-20 10:40 | 2025-04-20 10:45 | 300 | 107 | Assignment | Web Development | 2 | |  |  |  |  |  |  |  |  |  | session\_id |
|  |  |  |  |  |  |  |  |  |  | 1 |
|  |  |  |  |  |  |  |  |  |  | 1 |
|  |  |  |  |  |  |  |  |  |  |  |
| Note: course\_viewed event is not linked to a specific activity but rather to the course itself. When the course\_viewed event occurs, the user is on the course page. The user may then view an activity within that course (like an Assignment, Quiz, Forum, etc.). Activities like assignment\_viewed, quiz\_viewed, or forum\_viewed will be logged next as the user interacts with specific course components.  **Events Not Tied to Specific Activities:**   1. **course\_viewed**    * **Event Name**: \core\event\course\_viewed    * **Triggered When**: A user views or accesses a course page.    * **Description**: This event is associated with the course itself and does not correspond to any specific activity in the course. 2. **user\_loggedin**    * **Event Name**: \core\event\user\_loggedin    * **Triggered When**: A user successfully logs into the Moodle system.    * **Description**: This event occurs whenever a user logs into Moodle, regardless of the course or activity they are interacting with. 3. **user\_loggedout**    * **Event Name**: \core\event\user\_loggedout    * **Triggered When**: A user logs out of the system.    * **Description**: This event is triggered when a user logs out of Moodle, without being tied to a specific course or activity. 4. **~~user\_created~~**    * **~~Event Name~~**~~: \core\event\user\_created~~    * **~~Triggered When~~**~~: A new user account is created.~~    * **~~Description~~**~~: This event occurs when a new user is created in Moodle, typically by an administrator. It’s not linked to any specific course or activity.~~ 5. **~~user\_profile\_viewed~~**    * **~~Event Name~~**~~: \core\event\user\_profile\_viewed~~    * **~~Triggered When~~**~~: A user’s profile is viewed.~~    * **~~Description~~**~~: This event is triggered when a user's profile is viewed, regardless of which course or activity they are currently participating in.~~ 6. **~~user\_updated~~**    * **~~Event Name~~**~~: \core\event\user\_updated~~    * **~~Triggered When~~**~~: A user’s profile information is updated.~~    * **~~Description~~**~~: This event is triggered when a user's profile is updated, such as when their email or other personal details are modified.~~ 7. **~~enrol\_user~~**    * **~~Event Name~~**~~: \core\event\enrol\_user~~    * **~~Triggered When~~**~~: A user is enrolled in a course.~~    * **~~Description~~**~~: This event occurs when a user is enrolled in a course but is not tied to a specific activity within that course.~~ 8. **~~unenrol\_user~~**    * **~~Event Name~~**~~: \core\event\unenrol\_user~~    * **~~Triggered When~~**~~: A user is unenrolled from a course.~~    * **~~Description~~**~~: This event is triggered when a user is unenrolled from a course. Like enrol\_user, it’s not related to any specific activity.~~ 9. **~~course\_category\_created~~**    * **~~Event Name~~**~~: \core\event\course\_category\_created~~    * **~~Triggered When~~**~~: A new course category is created in the system.~~    * **~~Description~~**~~: This event is tied to the creation of course categories and does not refer to any specific course or activity.~~ 10. **~~course\_category\_updated~~**     * **~~Event Name~~**~~: \core\event\course\_category\_updated~~     * **~~Triggered When~~**~~: A course category is updated.~~     * **~~Description~~**~~: This event is triggered when a course category is updated, such as changing its name or description. It’s not related to any specific activity within the courses.~~ 11. **~~course\_category\_deleted~~**     * **~~Event Name~~**~~: \core\event\course\_category\_deleted~~     * **~~Triggered When~~**~~: A course category is deleted.~~     * **~~Description~~**~~: This event occurs when a course category is deleted from the system.~~ 12. **~~grade\_item\_created~~**     * **~~Event Name~~**~~: \core\event\grade\_item\_created~~     * **~~Triggered When~~**~~: A grade item (such as a gradebook entry) is created for a course.~~     * **~~Description~~**~~: This event is related to the creation of grade items, but it’s not tied to specific activities or actions taken within the course.~~ 13. **~~file\_uploaded~~**     * **~~Event Name~~**~~: \core\event\file\_uploaded~~     * **~~Triggered When~~**~~: A file is uploaded to the Moodle system.~~     * **~~Description~~**~~: This event occurs when any file is uploaded to Moodle, but it is not associated with a specific course or activity.~~ 14. **~~file\_deleted~~**     * **~~Event Name~~**~~: \core\event\file\_deleted~~     * **~~Triggered When~~**~~: A file is deleted from Moodle.~~     * **~~Description~~**~~: This event is triggered when a file is deleted from Moodle, regardless of the course or activity it was associated with.~~ 15. **~~user\_picture\_updated~~**     * **~~Event Name~~**~~: \core\event\user\_picture\_updated~~     * **~~Triggered When~~**~~: A user updates their profile picture.~~     * **~~Description~~**~~: This event happens when a user updates their profile picture and is not linked to a specific course or activity.~~ 16. **~~theme\_changed~~**     * **~~Event Name~~**~~: \core\event\theme\_changed~~     * **~~Triggered When~~**~~: The site theme is changed.~~     * **~~Description~~**~~: This event is triggered when the site’s theme is changed, not related to any course or activity.~~   Note: events not being used by students are crossed-out. |  |  |  |  |  |  |  |  |  | 1 |
|  |  |  |  |  |  |  |  |  |  | 2 |

/\*with duration, grouping by session, count of events in the session (count of events **matching the current** eventname in the **same session** for the user) and count of events in total (Count of events **matching the current** event name across **all sessions** for the user.) \*/

WITH EventSessions AS (

SELECT

l.userid,

l.eventname,

l.action,

l.target,

l.timecreated AS start\_time,

LEAD(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated) AS next\_event\_time,

TIMESTAMPDIFF(

SECOND,

FROM\_UNIXTIME(l.timecreated),

FROM\_UNIXTIME(

LEAD(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated)

)

) AS duration\_seconds,

cm.instance AS activity\_id,

m.name AS activity\_name,

c.fullname AS course\_name,

-- Generate session ID based on idle gaps greater than 3600 seconds (60 minutes)

SUM(

CASE

WHEN TIMESTAMPDIFF(SECOND,

FROM\_UNIXTIME(LAG(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated)),

FROM\_UNIXTIME(l.timecreated)) > 3600 THEN 1

ELSE 0

END

) OVER (PARTITION BY l.userid ORDER BY l.timecreated) + 1 AS session\_id

FROM

mdl\_logstore\_standard\_log l

JOIN

mdl\_course\_modules cm ON l.contextinstanceid = cm.id

JOIN

mdl\_modules m ON cm.module = m.id

JOIN

mdl\_course c ON cm.course = c.id

WHERE

l.userid IN (

SELECT u.id

FROM mdl\_user u

JOIN mdl\_user\_enrolments ue ON ue.userid = u.id

JOIN mdl\_enrol e ON e.id = ue.enrolid

JOIN mdl\_course c ON c.id = e.courseid

JOIN mdl\_role\_assignments ra ON ra.userid = u.id

JOIN mdl\_context ctx ON ctx.id = ra.contextid

JOIN mdl\_role r ON r.id = ra.roleid

WHERE u.department = 'ITfu-23'

AND r.shortname = 'student'

AND ctx.contextlevel = 50

)

AND l.eventname != ''

AND l.timecreated > 0

),

EventCounts AS (

SELECT

userid,

eventname,

session\_id,

COUNT(\*) AS event\_count\_in\_session

FROM EventSessions

GROUP BY userid, eventname, session\_id

),

EventTotals AS (

SELECT

userid,

eventname,

COUNT(\*) AS event\_count\_total

FROM EventSessions

GROUP BY userid, eventname

)

SELECT

es.userid,

es.eventname,

es.action,

es.target,

es.start\_time,

es.next\_event\_time,

es.duration\_seconds,

es.activity\_id,

es.activity\_name,

es.course\_name,

es.session\_id,

ec.event\_count\_in\_session,

et.event\_count\_total

FROM EventSessions es

LEFT JOIN EventCounts ec

ON es.userid = ec.userid

AND es.eventname = ec.eventname

AND es.session\_id = ec.session\_id

LEFT JOIN EventTotals et

ON es.userid = et.userid

AND es.eventname = et.eventname

ORDER BY

es.userid,

es.session\_id,

es.start\_time ASC;

Exanple of data to be returned:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **userid** | **eventname** | **action** | **target** | **start\_time** | **next\_event\_time** | **duration\_seconds** | **activity\_id** | **activity\_name** | **course\_name** | **session\_id** | **event\_count\_in\_session** | **event\_count\_total** |
| 1 | \mod\_quiz\event\attempt\_started | started | attempt | 1610000000 | 1610000500 | 500 | 101 | Quiz 1 | Course A | 1 | 1 | 2 |
| 1 | \mod\_quiz\event\attempt\_submitted | submitted | attempt | 1610000500 | 1610001100 | 600 | 101 | Quiz 1 | Course A | 1 | 1 | 1 |
| 1 | \mod\_quiz\event\attempt\_started | started | attempt | 1610001100 |  |  | 101 | Quiz 1 | Course A | 2 | 1 | 2 |
| 2 | \mod\_forum\event\discussion\_viewed | viewed | discussion | 1610000000 | 1610000600 | 600 | 202 | Forum A | Course B | 1 | 1 | 1 |
| 2 | \mod\_quiz\event\attempt\_started | started | attempt | 1610000600 | 1610001200 | 600 | 101 | Quiz 1 | Course B | 1 | 2 | 2 |
| 2 | \mod\_quiz\event\attempt\_started | started | attempt | 1610001200 |  |  | 101 | Quiz 1 | Course B | 2 | 1 | 2 |

### /\* Additions:

1. EventDurationStats: calculates the average (AVG) and standard deviation (STDDEV) of durations for each event, per user, per session.
2. EventSessionFrequency: calculates the frequency of each event per session and per user:
   * frequency\_event\_per\_session: this represents how often an event appears in a session for a user.
   * frequency\_event\_total: this represents how often an event appears across all sessions for a user.
3. The final SELECT includes these newly calculated values, such as the average and standard deviation of durations, as well as event frequencies.

The results will now include the requested information:

* Count of the same event per session per user
* Average duration of the same event per session per user
* Standard deviation of durations for the same event per session per user
* Total count of the same event across all sessions for the user
* Frequency of the event per session per user
* Frequency of the event across all sessions for the user

 **How many times the same event (by eventname) appears per session per user** is already covered by event\_count\_in\_session.

 **Average duration for the same event (by eventname) per session per user** will be calculated using AVG(duration\_seconds) for each user and session.

 **Standard deviation of durations for the same event (by eventname) per session per user** will be calculated using STDDEV(duration\_seconds) for each user and session.

 **Total times the same event (by eventname) occurred across all sessions of the user** is covered by event\_count\_total.

 **How often this event type (the same event) appears per session per user** will be calculated as event\_count\_in\_session / total\_events\_in\_session.

 **How often this event type appears across all the user’s sessions per user** will be calculated as event\_count\_total / total\_events\_for\_user.

### \*/

### WITH EventSessions AS (

### SELECT

### l.userid,

### l.eventname,

### l.action,

### l.target,

### l.timecreated AS start\_time,

### LEAD(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated) AS next\_event\_time,

### TIMESTAMPDIFF(

### SECOND,

### FROM\_UNIXTIME(l.timecreated),

### FROM\_UNIXTIME(

### LEAD(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated)

### )

### ) AS duration\_seconds,

### cm.instance AS activity\_id,

### m.name AS activity\_name,

### c.fullname AS course\_name,

### -- Generate session ID based on idle gaps greater than 3600 seconds (60 minutes)

### SUM(

### CASE

### WHEN TIMESTAMPDIFF(SECOND,

### FROM\_UNIXTIME(LAG(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated)),

### FROM\_UNIXTIME(l.timecreated)) > 3600 THEN 1

### ELSE 0

### END

### ) OVER (PARTITION BY l.userid ORDER BY l.timecreated) + 1 AS session\_id

### FROM

### mdl\_logstore\_standard\_log l

### JOIN

### mdl\_course\_modules cm ON l.contextinstanceid = cm.id

### JOIN

### mdl\_modules m ON cm.module = m.id

### JOIN

### mdl\_course c ON cm.course = c.id

### WHERE

### l.userid IN (

### SELECT u.id

### FROM mdl\_user u

### JOIN mdl\_user\_enrolments ue ON ue.userid = u.id

### JOIN mdl\_enrol e ON e.id = ue.enrolid

### JOIN mdl\_course c ON c.id = e.courseid

### JOIN mdl\_role\_assignments ra ON ra.userid = u.id

### JOIN mdl\_context ctx ON ctx.id = ra.contextid

### JOIN mdl\_role r ON r.id = ra.roleid

### WHERE u.department = 'ITfu-23'

### AND r.shortname = 'student'

### AND ctx.contextlevel = 50

### )

### AND l.eventname != ''

### AND l.timecreated > 0

### ),

### EventCounts AS (

### SELECT

### userid,

### eventname,

### session\_id,

### COUNT(\*) AS event\_count\_in\_session

### FROM EventSessions

### GROUP BY userid, eventname, session\_id

### ),

### EventTotals AS (

### SELECT

### userid,

### eventname,

### COUNT(\*) AS event\_count\_total

### FROM EventSessions

### GROUP BY userid, eventname

### ),

### EventDurationStats AS (

### SELECT

### userid,

### eventname,

### session\_id,

### AVG(duration\_seconds) AS avg\_duration\_per\_session,

### STDDEV(duration\_seconds) AS stddev\_duration\_per\_session

### FROM EventSessions

### GROUP BY userid, eventname, session\_id

### ),

### EventSessionFrequency AS (

### SELECT

### es.userid,

### es.eventname,

### es.session\_id,

### ec.event\_count\_in\_session,

### et.event\_count\_total,

### eds.avg\_duration\_per\_session,

### eds.stddev\_duration\_per\_session,

### ROUND(ec.event\_count\_in\_session \* 1.0 / (SELECT COUNT(\*) FROM EventSessions WHERE userid = es.userid AND session\_id = es.session\_id), 4) AS frequency\_event\_per\_session,

### ROUND(et.event\_count\_total \* 1.0 / (SELECT COUNT(\*) FROM EventSessions WHERE userid = es.userid), 4) AS frequency\_event\_total

### FROM EventSessions es

### LEFT JOIN EventCounts ec ON es.userid = ec.userid AND es.eventname = ec.eventname AND es.session\_id = ec.session\_id

### LEFT JOIN EventTotals et ON es.userid = et.userid AND es.eventname = et.eventname

### LEFT JOIN EventDurationStats eds ON es.userid = eds.userid AND es.eventname = eds.eventname AND es.session\_id = eds.session\_id

### )

### SELECT

### es.userid,

### es.eventname,

### es.action,

### es.target,

### es.start\_time,

### es.next\_event\_time,

### es.duration\_seconds,

### es.activity\_id,

### es.activity\_name,

### es.course\_name,

### es.session\_id,

### ec.event\_count\_in\_session,

### eds.avg\_duration\_per\_session,

### eds.stddev\_duration\_per\_session,

### et.event\_count\_total,

### es.frequency\_event\_per\_session,

### es.frequency\_event\_total

### FROM EventSessions es

### LEFT JOIN EventCounts ec

### ON es.userid = ec.userid

### AND es.eventname = ec.eventname

### AND es.session\_id = ec.session\_id

### LEFT JOIN EventTotals et

### ON es.userid = et.userid

### AND es.eventname = et.eventname

### LEFT JOIN EventDurationStats eds

### ON es.userid = eds.userid

### AND es.eventname = eds.eventname

### AND es.session\_id = eds.session\_id

### ORDER BY

### es.userid,

### es.session\_id,

### es.start\_time ASC;

### Example of data to be returned:

### 

### Approach to the creation of the data set:

### Aggregate the data to the session level:

* **Session id**
* **How many times the same event (by eventname) appears per session per user** is already covered by event\_count\_in\_session.
* **Average duration for the same event (by eventname) per session per user** will be calculated using AVG(duration\_seconds) for each user and session.
* **Standard deviation of durations for the same event (by eventname) per session per user** will be calculated using STDDEV(duration\_seconds) for each user and session.
* **How often this event type (the same event) appears per session per user** will be calculated as event\_count\_in\_session / total\_events\_in\_session;

### For each event, 4 feature columns (ref. 1.) should be created;

### Label each session record based on the mapping “Mapping of events to learning styles by event“.

### Use MinMaxScaler or StandardScaler to ensure equal feature importance (make the loss function to behave evenly);

### Peform exploratory data analysis, model training.

**/\*Version of the query without using** LEAD(), using a **correlated subquery** instead. This version may be **slower** on large datasets, since it runs a subquery for every log row. It is possible to limit it further with a WHERE clause on l.timecreated to improve performance.\*/

SELECT

l.userid,

l.eventname,

l.action,

l.target,

l.timecreated AS start\_time,

(

SELECT MIN(l2.timecreated)

FROM mdl\_logstore\_standard\_log l2

WHERE l2.userid = l.userid

AND l2.timecreated > l.timecreated

) AS next\_event\_time,

TIMESTAMPDIFF(

SECOND,

FROM\_UNIXTIME(l.timecreated),

FROM\_UNIXTIME(

(

SELECT MIN(l2.timecreated)

FROM mdl\_logstore\_standard\_log l2

WHERE l2.userid = l.userid

AND l2.timecreated > l.timecreated

)

)

) AS duration\_seconds,

cm.instance AS activity\_id,

m.name AS activity\_name,

c.fullname AS course\_name

FROM

mdl\_logstore\_standard\_log l

JOIN

mdl\_course\_modules cm ON l.contextinstanceid = cm.id

JOIN

mdl\_modules m ON cm.module = m.id

JOIN

mdl\_course c ON cm.course = c.id

WHERE

l.userid IN (

SELECT u.id

FROM mdl\_user u

JOIN mdl\_user\_enrolments ue ON ue.userid = u.id

JOIN mdl\_enrol e ON e.id = ue.enrolid

JOIN mdl\_course c ON c.id = e.courseid

JOIN mdl\_role\_assignments ra ON ra.userid = u.id

JOIN mdl\_context ctx ON ctx.id = ra.contextid

JOIN mdl\_role r ON r.id = ra.roleid

WHERE u.department = 'ITfu-23'

AND r.shortname = 'student'

AND ctx.contextlevel = 50

)

AND l.eventname != ''

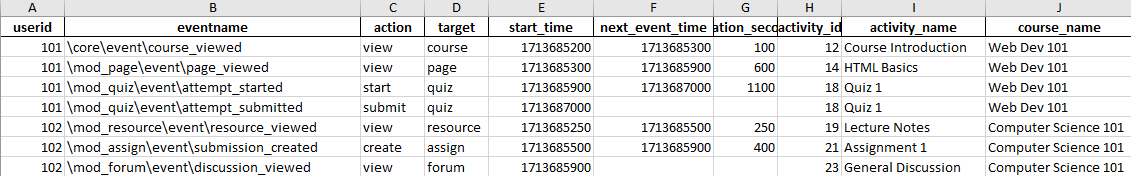
AND l.timecreated > 0

ORDER BY

l.userid,

l.timecreated ASC;

Example of data to be returned:



/\* Group Moodle log events into **sessions based on idle gaps** (e.g., 60 minutes or 3600 seconds (group into sessions based on idle gaps)): the logic compares each event's timestamp to the previous event for the same user and increments a **session counter** when the time gap exceeds the threshold\*/

WITH ordered\_logs AS (

SELECT

l.userid,

l.eventname,

l.action,

l.target,

l.timecreated,

cm.instance AS activity\_id,

m.name AS activity\_name,

c.fullname AS course\_name,

LAG(l.timecreated) OVER (PARTITION BY l.userid ORDER BY l.timecreated) AS prev\_time

FROM

mdl\_logstore\_standard\_log l

JOIN

mdl\_course\_modules cm ON l.contextinstanceid = cm.id

JOIN

mdl\_modules m ON cm.module = m.id

JOIN

mdl\_course c ON cm.course = c.id

WHERE

l.userid IN (

SELECT u.id

FROM mdl\_user u

JOIN mdl\_user\_enrolments ue ON ue.userid = u.id

JOIN mdl\_enrol e ON e.id = ue.enrolid

JOIN mdl\_course c ON c.id = e.courseid

JOIN mdl\_role\_assignments ra ON ra.userid = u.id

JOIN mdl\_context ctx ON ctx.id = ra.contextid

JOIN mdl\_role r ON r.id = ra.roleid

WHERE u.department = 'ITfu-23'

AND r.shortname = 'student'

AND ctx.contextlevel = 50

)

AND l.eventname != ''

AND l.timecreated > 0

),

session\_marks AS (

SELECT \*,

CASE

WHEN prev\_time IS NULL THEN 1

WHEN l.timecreated - prev\_time > 3600 THEN 1

ELSE 0

END AS new\_session\_flag

FROM ordered\_logs l

),

session\_numbered AS (

SELECT \*,

SUM(new\_session\_flag) OVER (PARTITION BY userid ORDER BY timecreated ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS session\_id

FROM session\_marks

)

SELECT

userid,

eventname,

action,

target,

timecreated,

activity\_id,

activity\_name,

course\_name,

session\_id

FROM session\_numbered

ORDER BY userid, timecreated;

Note:

